ELSEVIER

Journal of Computational and Applied Mathematics 74 (1996) 33–50

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

The stabilized *V*-cycle method

O. Axelsson

Faculty of Mathematics and Informatics, University of Nijmegen, Toernooiveld, 6525ED Nijmegen, Netherlands

Received 30 June 1995

Abstract

Algebraic multilevel iterations methods are preconditioning algorithms, to solve elliptic type partial differential equations by iteration which can give both a robust and optimal, or nearly optimal, convergence rate and require per iteration step an arithmetic complexity proportional to the degree of freedoms in the problem. In addition, each iteration step can, in general, be implemented efficiently on massively parallel computers. To stabilize the condition number in the *V*-cycle version of the method one can use polynomial stabilization or inner solutions at certain, properly chosen levels in the multilevel hierarchy of meshes. The latter is considered here.

Keywords: Multilevel method, Stabilization, Finite element method, *V*-cycle

AMS classification: 65F10, 65N20, 65N30; CR:G1.3

1. Introduction

Over the past few years, *multilevel* methods have become a common denominator for a broad class of iterative solution methods. All such methods work recursively on a sequence of node levels, which are either defined geometrically from a finite difference or finite element mesh or defined via a matrix graph. They have the favorable property of yielding an optimal or close to optimal rate of convergence under quite general conditions while the computational complexity per node point and iteration cycle is bounded independently of the number of levels.

A classical type of multilevel method is the multigrid method for difference or finite element matrices of elliptic type, originally presented in [25, 30, 31] and later significantly extended in [28, 32, 33] and elsewhere. Based on a sequence of (normally) nested meshes, one defines here smoothing operators on each level and prolongation and restriction operators between consecutive levels.

Other types of multilevel iteration methods are the methods based on the recursive use of two-level finite element meshes and associated matrix blocks, using either standard basis functions or hierarchical basis functions.

* E-mail: axelsson@sci.kun.nl.

PII S0377-0427(96)00016-7

Such methods were originally studied in the two-(or few-) level context in [1, 13, 26] and were later extended to a full hierarchical basis function formulation in [41] and in a series of papers by the author and Vassilevski [22–24] to a stabilized version. In these latter papers the condition number was stabilized (bounded) to be independent of the number of levels by use of certain polynomials approximating the arising Schur complement matrices (see [22, 23], and also [35, 36] for a similar approach). The matrix so constructed (or rather algorithm to compute the action of the implicitly defined discrete operator) was used as a preconditioner, in a preconditioned conjugate gradient method, for instance.

In this context and for a later reference it should be noted that the hierarchical basis function method results in a preconditioning matrix for which the condition number grows as $O(\log h^{-1})^2$ and $O(h^{-1})$ in two-dimensional and three-dimensional mesh problems, respectively.

For a survey of such and other methods see [21]. For a recent exposition of multigrid methods in a finite element context, see [27].

A third type of multilevel iteration method to construct a preconditioner has been based on a sequence of node sets defined from the matrix graph, associated with the given matrix, see [5, 19]. This type of method can be seen as an incomplete block matrix factorization method (see [2, 7, 11, 29]). Indeed, each action of the preconditioner involves a forward block matrix recursion, a solution of a system for the coarse level matrix and a backward block matrix recursion. In the polynomially stabilized version these recursions are repeated on each level a number of times (equal to the polynomial degree minus one). However, instead of the more common "line blocks" in the publications on block matrix factorizations, in the multilevel context one uses a global partitioning of the node points, which is crucial for achieving a stabilized, or close to stabilized condition number.

A particular example of such a global ordering is the recursive use of the familiar red-black ordering for a five-point or nine-point difference mesh (see [3, 12]).

Given a nested sequence of node points $\{\Omega_k\}$ on each level in such methods the node set is split into two parts $\Omega_k \backslash \Omega_{k-1}$ and Ω_{k-1}. The matrix block corresponding to the first set acts as a smoothing operator and the two matrix blocks which define the coupling between the two sets define "restriction" and "prolongation" operators. As for the corresponding finite element version one can use matrix polynomials to approximate the arising Schur complements in order to stabilize the condition number.

However, using the same order of the polynomials on all levels does not always yield a method of optimal computational complexity per iteration cycle, but it was shown in [39, 40], see also [5, 12, 19], that by use of polynomial stabilization at only some levels with a polynomial of a sufficiently large degree, the method regained its optimal order of computational complexity.

The above-mentioned finite element multilevel iteration method [22, 23] and the algebraic matrix graph based method have both been called AMLI-algebraic multilevel iteration methods. It is proposed here to rename the finite element version to FEMLI and keep the name AMLI for the matrix graph version.

In practical implementations it has been found that the polynomially stabilized methods require much recursive overhead and, in addition, interprocessor communication overhead when implemented on (massively) parallel computers with distributed memory. Therefore, the simplest version of the multilevel iteration method, the V-cycle version, where each iteration cycle (or one action of the preconditioner) involves only a single forward block matrix recursion, followed by a solution once

of the coarse mesh system and a single backward block matrix recursion, is most interesting in particular for parallel computers.

However, the condition number for this version can grow significantly with the number of levels. Therefore, in a series of papers [16–18], a short level version of the V-cycle AMLI method was studied. Because only few levels are used, the condition number will not be large. By properly relating the size of the coarse mesh to the fine mesh, depending on the solution method used on the coarse mesh, one could limit the computational complexity per iteration cycle to be proportional to the total degree of freedom. In addition, it was found that the method could be implemented efficiently on massively parallel computers with asymptotically optimal speedup or parallel efficiency when the number of processors was properly balanced to the number of nodepoints.

The spectral condition number for the corresponding preconditioned matrix was not fully studied in these papers. In the present paper we show that for the FEMLI method the condition number gets essentially stabilized when the short version of the method is used. The stabilization is stronger for spatially three-dimensional than for two-dimensional problems. This holds for quite general classes of problems and for both h- and p-versions of finite element methods. In the present paper we consider the h-version using piecewise linear basis functions for a triangulation of a polygonal domain Ω. The technique can be extended straightforwardly to three-dimensional (polytope) problems. The problem we consider is the Poisson problem (here shown for $\Omega \subset \mathbb{R}^2$)

$$\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(b\frac{\partial u}{\partial y}\right) = f(x, y), \quad (x, y) \in \Omega, \tag{1.1}$$

with standard types of boundary conditions. Ω is a bounded domain. Here a and b are assumed (for simplicity of presentation) to be piecewise positive constants with no discontinuities within the elements of the coarsest mesh. They can be discontinuous between elements on the coarsest mesh level. Since in our short level version, the coarsest mesh will actually be quite a fine mesh itself, as we shall see, this is no serious restriction, i.e., we can permit a significant number of jumps in the coefficients. (Incidently, this indicates that there is little need for the so-called homogenization techniques.)

The bilinear form for (1.1) takes the form

$$a(u, v) = \int_\Omega \left(a\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + b\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right) \mathrm{d}x\, \mathrm{d}y \tag{1.2}$$

plus a proper boundary integral term in case there are inhomogeneous boundary conditions of non-Dirichlet type. However, for notational simplicity, we assume homogeneous boundary conditions.

Given a finite element subspace V_l of $\mathring{H}^1(\Omega)$, our problem is then to find u_h, such that

$$a(u_h, v_h) = \int_\Omega f v_h, \qquad \text{for all } v_h \in V_l.$$

The FEMLI method is applicable for still more general finite element problems such as for convection-diffusion problems. However, it can be difficult to construct a spectral theory for this case, so we limit the presentation here to the symmetric and coercive bilinear form (1.2).

2. The condition number of the short AMLI method

Consider a sequence of nested finite element meshes

$$\Omega_{k_0} \subset \Omega_{k_0+1} \subset \cdots \subset \Omega_{l-1} \subset \Omega_l, \tag{2.1}$$

where Ω_{k_0} is the coarsest and Ω_l the finest mesh. Let $\{V_k\}_{k=k_0}^{l}$ be the corresponding finite element function spaces for an h-version of finite element method. On each mesh, the node set is partitioned into two sets,

$$\Omega_{k-1} \text{ and } \Omega_k \backslash \Omega_{k-1}, \quad k_0 + 1 \leqslant k \leqslant l.$$

To each mesh a matrix $A^{(k)}$ is defined which can be either the finite element stiffness matrix corresponding to V_k and the bilinear form (1.2) (see [1, 4, 13, 23]), or a matrix computed algebraically as in a block incomplete factorization method starting with the finite element matrix $A^{(l)}$ and recursively partitioning $A^{(k)}$ in 2×2 blocks corresponding to a node point ordering defined by (2.1), see [5, 12, 19].

Similarly, for the finite element matrix sequence, $A^{(k)}$ is partitioned into 2×2 blocks corresponding to the finite element subspaces $V_k \backslash V_{k-1}$ and V_{k-1},

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix} \begin{matrix} \} V_k \backslash V_{k-1} \\ \} V_{k-1} \end{matrix}.$$

In this paper we shall consider only the latter method. Let $M^{(k_0)}$ be given a sequence of preconditioners $\{M^{(k)}\}$ to $\{A^{(k)}\}$ is defined recursively from the coarsest mesh to the finest as

$$M^{(k)} = \begin{bmatrix} B_{11}^{(k)^{-1}} & 0 \\ A_{21}^{(k)} & I_2^{(k)} \end{bmatrix} \begin{bmatrix} I_1^{(k)} & B_{11}^{(k)} A_{12}^{(k)} \\ 0 & \alpha_{k-1} M^{(k-1)} \end{bmatrix}, \quad k = k_0 + 1, \ldots, l, \tag{2.2}$$

where $B_{11}^{(k)}$ is an approximation of $A_{11}^{(k)^{-1}}$ and $I_1^{(k)}$, $I_2^{(k)}$ are identity matrices for the node sets $\Omega_k \backslash \Omega_{k-1}$ and Ω_{k-1}, respectively. Here the sequence α_k, $\alpha_k \geqslant 1$, will be determined later. Also, for later use, let λ_k denote the largest eigenvalue of $A^{(k)^{-1}} M^{(k)}$.

It follows from (2.2) that the preconditioner $M^{(l)}$ can be written in explicit form as a block matrix (with $(l-k_0+1) \times (l-k_0+1)$ blocks). However, to compute the action of $M^{(k)^{-1}}$ it is more efficient to use the factorized and recursive form (2.2). This involves $l-k_0$ forward substitution steps, a solution with matrix $M^{(k_0)}$ and $l-k_0$ backward substitution steps as for a block matrix factorization method. In these steps, only matrix times vector operations are involved. This structure is somewhat similar to the structure of a V-cycle in a multigrid method with $B_{11}^{(k)}$ corresponding to the action of a smoothing operator and $A_{21}^{(k)}, A_{12}^{(k)}$ corresponding to restriction and prolongation operators, respectively. However, in multigrid methods one works on the whole space V_k during the smoothing operation and the restriction and prolongation operators *must be defined* as operators between the spaces V_k and V_{k-1}.

The solution method for the matrix system on the coarsest level can be a direct solution method. However, normally, it is more efficient to use an (inner) iteration method, frequently with a preconditioner of an easy form such as in a (block) diagonal matrix in an additive domain decomposition method or an element by element preconditioner.

We shall assume that

$$\beta_k A_{11}^{(k)^{-1}} \leqslant B_{11}^{(k)} \leqslant A_{11}^{(k)^{-1}} \tag{2.3}$$

(in a positive semidefinite sense), where $0 < \beta_k \leqslant 1$ and that $M^{(k_0)}$ is such that

$$A^{(k_0)} \leqslant M^{(k_0)} \leqslant \hat{\lambda}_0 A^{(k_0)} \tag{2.4}$$

for some number $\hat{\lambda}_0 \geqslant 1$, which does not depend on k_0. To simplify the presentation we assume also that $B_{11}^{(k)}$ is a sufficiently accurate approximation of $A_{11}^{(k)^{-1}}$ so that β_k is sufficiently close to unity for condition (2.11) below to hold.

In order to analyze the condition number of $M^{(k)^{-1}}A^{(k)}$, we will use the form

$$M^{(k)} = \begin{bmatrix} B_{11}^{(k)^{-1}} & A_{12}^{(k)} \\ A_{21}^{(k)} & \alpha_{k-1}M^{(k-1)} + A_{21}^{(k)}B_{11}^{(k)}A_{12}^{(k)} \end{bmatrix}.$$

Hence,

$$M^{(k)} - A^{(k)} = \begin{bmatrix} B_{11}^{(k)^{-1}} - A_{11}^{(k)} & 0 \\ 0 & \alpha_{k-1}M^{(k-1)} - (A_{22}^{(k)} - A_{21}^{(k)}B_{11}^{(k)}A_{12}^{(k)}) \end{bmatrix}. \tag{2.5}$$

In order to derive lower and upper bounds of the eigenvalues of $M^{(k)^{-1}}A^{(k)}$ we notice first that (2.3) implies that

$$\beta_k^{-1}A_{11}^{(k)} \geqslant B_{11}^{(k)^{-1}} \geqslant A_{11}^{(k)}. \tag{2.6}$$

Now α_{k-1} is determined such that the lower bound is unity, i.e.,

$$\alpha_{k-1}M^{(k-1)} \geqslant A_{22}^{(k)} - A_{21}^{(k)}B_{11}^{(k)}A_{12}^{(k)}. \tag{2.7}$$

This is equivalent to

$$\alpha_{k-1}M^{(k-1)} \geqslant S_2^{(k)} + A_{21}^{(k)}(A_{11}^{(k)^{-1}} - B_{11}^{(k)})A_{12}^{(k)},$$

where

$$S_2^{(k)} = A_{22}^{(k)} - A_{21}^{(k)}A_{11}^{(k)^{-1}}A_{12}^{(k)},$$

which is the Schur complement to $A^{(k)}$ w.r.t. $A_{11}^{(k)}$. Further, we shall use the following Lemmas.

A classical result in Linear Algebra (see [7], for instance) states:

Lemma 2.1. *Let* $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ *be symmetric and positive definite. Then*

$$\inf_{x_j} \left\{ (x_1^{\mathrm{T}}, x_2^{\mathrm{T}}) \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\} = x_i^{\mathrm{T}} S_i x_i,$$

$i \neq j$, $i,j = 1,2$, *where* $S_i = A_{ii} - A_{ij}A_{jj}^{-1}A_{ji}$.

Lemma 2.2 (Axelsson [1] and Axelsson and Gustafisson [13]). *Let*

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix} \begin{matrix} \} V_k \backslash V_{k-1} \\ \} V_{k-1} \end{matrix}$$

be a stiffness finite element matrix for the bilinear form (1.2) *and finite element subspace* V_k. *Then*

$$A^{(k-1)} \geqslant S_2^{(k)} \geqslant (1-\gamma_k^2)A^{(k-1)},$$

where γ_k, $0 < \gamma_k < 1$ *is the constant in the extended Cauchy–Schwarz–Bunyakowski inequality,*

$$a(u_1^{(k)}, u_2^{(k)}) \leqslant \gamma_k \{a(u_1^{(k)}, u_1^{(k)}) a(u_2^{(k)}, u_2^{(k)})\}^{1/2}, \tag{2.8}$$

where $u_1^{(k)} \in V_k \backslash V_{k-1}$, $u_2^{(k)} \in V_{k-1}$.

As shown in [1, 13, 26], the constant γ_k is strictly less than unity, uniformly in the mesh-size parameter or, equivalently, in the level number. Further, γ_k depends on the angles in the triangulation (see [37]) and for a congruent recursive triangulation, the angles remain the same, so γ_k does not depend on k, for such meshes. Now using (2.5) we find

$$0 \leqslant \frac{x^{(k)^T}(M^{(k)} - A^{(k)})x^{(k)}}{x^{(k)^T}A^{(k)}x^{(k)}}$$

$$\leqslant \frac{x_1^{(k)^T}(B_{11}^{(k)^{-1}} - A_{11}^{(k)})x_1^{(k)} + x_2^{(k)^T}(\alpha_{k-1}M^{(k-1)} - A_{22}^{(k)} + A_{21}^{(k)}B_{11}^{(k)}A_{12}^{(k)})x_2^{(k)}}{x^{(k)^T}A^{(k)}x^{(k)}},$$

when

$$x^{(k)} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix}$$

is partitioned consistently with the matrix partitioning.
Using (2.6) and Lemma 2.1 we find

$$0 \leqslant \frac{x^{(k)^T}(M^{(k)} - A^{(k)})x^{(k)}}{x^{(k)^T}A^{(k)}x^{(k)}}$$

$$\leqslant \max\left\{ \sup_{x_1^{(k)}} \frac{(\beta_k^{-1} - 1)x_1^{(k)^T}A_{11}^{(k)}x_1^{(k)}}{x_1^{(k)^T}S_1^{(k)}x_1^{(k)}}, \sup_{x_2^{(k)}} \frac{x_2^{(k)^T}(\alpha_{k-1}M^{(k-1)} - A_{22}^{(k)} + A_{21}^{(k)}B_{11}^{(k)}A_{12}^{(k)})x_2^{(k)}}{x_2^{(k)^T}S_2^{(k)}x_2^{(k)}} \right\}. \tag{2.9}$$

To find an upper bound we rewrite the rightmost expression in (2.9) as

$$\frac{x_2^{(k)^T}(\alpha_{k-1}M^{(k-1)} - S_2^{(k)} + A_{21}^{(k)}(B_{11}^{(k)} - A_{11}^{(k)^{-1}})A_{12}^{(k)})x_2^{(k)}}{x_2^{(k)^T}S_2^{(k)}x_2^{(k)}}.$$

Using the assumption (2.3) and Lemma 2.2, this shows that

$$\frac{x^{(k)^T}(M^{(k)}-A^{(k)})x^{(k)}}{x^{(k)^T}A^{(k)}x^{(k)}} \leqslant \max\left\{\frac{\beta_k^{-1}-1}{1-\delta_k^2},\right.$$

$$\left.\sup_{x_2^{(k)}}\frac{x_2^{(k)^T}\alpha_{k-1}M^{(k-1)}x_2^{(k)}}{(1-\gamma_k^2)x_2^{(k)^T}A^{(k-1)}x_2^{(k)}}-1\right\} = \max\left\{\frac{\beta_k^{-1}-1}{1-\delta_k^2},\frac{\alpha_{k-1}\lambda_{k-1}}{1-\gamma_k^2}-1\right\}, \tag{2.10}$$

where

$$\frac{1}{1-\delta_k^2} = \sup_{x_1^{(k)}}\frac{x_1^{(k)^T}A_{11}^{(k)}x_1^{(k)}}{x_1^{(k)^T}S_1^{(k)}x_1^{(k)}}.$$

As it turns out (see [9, 13]), the computation of both γ_k and δ_k can occur on element by element level and by taking the maximum of the element-wise values.

We assume that the approximation $B_{11}^{(k)}$ of $A_{11}^{(k)^{-1}}$ is sufficiently accurate so that

$$\frac{\beta_k^{-1}-\delta_k^2}{1-\delta_k^2} \leq \frac{\alpha_{k-1}\lambda_{k-1}}{1-\gamma_k^2}. \tag{2.11}$$

(For instance, $B_{11}^{(k)}$ can be computed as a sufficiently accurate approximate inverse, as shown in [34]; see also [7]). It follows then from (2.10) that the following recursive relation holds:

$$\lambda_k \leqslant \frac{1}{1-\gamma_k^2}\alpha_{k-1}\lambda_{k-1}$$

and

$$\lambda_l = \prod_{k+1}^{l}\left(\frac{\alpha_{k-1}}{1-\gamma_k^2}\right)\hat{\lambda}_0.$$

Let

$$\sigma = \sigma_{l,k_0} = \left\{\prod_{k_0+1}^{l}\frac{\alpha_{k-1}}{1-\gamma_k^2}\right\}^{1/l-k_0}.$$

Then

$$\lambda_l = \sigma^{l-k_0}\hat{\lambda}_0, \quad l \geqslant k_0.$$

Note that λ_l is an upper bound of the condition number of $M^{(l)^{-1}}A^{(l)}$, which we denote by κ_{l,k_0} since it depends on both l and k_0,

$$\kappa_{l,k_0} = \sigma_{l,k_0}^{l-k_0}\hat{\lambda}_0. \tag{2.12}$$

We summarize the main result of this section. Let

$$\lambda_k = \max_{x^{(k)}}\frac{x^{(k)^T}M^{(k)}x^{(k)}}{x^{(k)^T}A^{(k)}x^{(k)}}.$$

Then (2.10) shows that

$$\lambda_k \leqslant \max\left\{\frac{\beta_k^{-1}-\delta_k^2}{1-\delta_k^2}, \frac{\alpha_{k-1}}{1-\gamma_k^2}\lambda_{k-1}\right\}, \quad k \geqslant k_0+1, \qquad \lambda_{k_0} = \hat{\lambda}_0. \tag{2.13}$$

3. Computational complexity

We consider now the computational complexity of the short FEMLI algorithm.
Let n_k denote the degrees of freedom (d.o.f.) on the mesh Ω_k on level k and assume for simplicity that all matrix–vector product operations have a computational complexity per d.o.f., in total bounded by a constant C, for each level. This means that all matrices involved have a bounded sparsity pattern, which, as well known, is a property which holds for the standard basis function matrices in the finite element method. In C we include also the vector operations during each visit on a new mesh level. Then, it is readily seen that the total computational complexity during an iteration step, which includes an application of the V-cycle form of the FEMLI preconditioner and other vector operations in the iterative (acceleration) method used, becomes

$$w_l \leqslant C(n_l + n_{l-1} + \cdots + n_{k_0+1}) + C_0 m_{k_0} n_{k_0}. \tag{3.1}$$

$C_0 m_{k_0}$ stands for the computational complexity per d.o.f. on the coarsest level. Here m_{k_0} can be the number of iterations done on this level, for instance, and C_0 depends then on the sparsity of the matrix involved on the coarse level.

Assume now that the degrees of freedom are related in a geometric progression, i.e., it holds

$$\frac{n_{k-1}}{n_k} \leqslant \rho < 1.$$

Typical values of ρ are $\rho = \frac{1}{4}$, which holds when each triangle in the mesh refinement process is divided in four congruent parts and $\rho = \frac{1}{2}$ which holds in the case of bisection of triangles. The latter is a generalization to general triangulations of the recursive red-black ordering used for finite difference meshes, see[3, 6]. For a three-dimensional problem $\rho = \frac{1}{8}$ and $\rho = \frac{1}{2}$, respectively.
It follows then from (3.1) that

$$\frac{w_l}{n_l} \leqslant C\frac{1-\rho^{l-k_0+1}}{1-\rho} + C_0 m_{k_0}\rho^{l-k_0} \leqslant \frac{C}{1-\rho} + C_0 m_{k_0}\rho^{l-k_0}.$$

The total computational complexity per d.o.f. in the short FEMLI preconditioned conjugate gradient method is then

$$\frac{W_l}{n_l} \leqslant \tfrac{1}{2} C \ln\tfrac{2}{\varepsilon}\sqrt{\kappa_{l,k_0}}\left(\tfrac{1}{1-\rho} + \tfrac{C_0}{C} m_{k_0}\rho^{l-k_0}\right), \tag{3.2}$$

where we have used the standard upper bound $\frac{1}{2}\sqrt{\kappa_{l,k_0}}\ln(2/\varepsilon)$ of the number of (outer) CG iterations, i.e., number of V-cycle steps. Here ε is the required relative tolerance of the $(A^{(l)})^{1/2}$-norm of the iteration errors. We want to relate k_0 to l so that the total computational complexity in (3.2) is

minimized. To this end we must consider some specific classes of solution methods on the coarse mesh.

Let $h_k = 2^{-k/d}$ in case of bisections (for the two-dimensional case, see [6] and for the three-dimensional case, see [15]) and $h_k = 2^{-k}$ in case of congruent divisions of triangles or tetrahedrons ($d = 2$ or 3, respectively) be the average mesh-size parameter of mesh Ω_k. We consider two cases, both of which are useful in practice:

(a) A blockdiagonal preconditioner, for instance an additive domain decomposition method or an element by element preconditioner for which the number of iterations on the coarse mesh varies as $m_{k_0} = ch_{k_0}^{-1}$, for some constant c.

(b) A block matrix incomplete factorization method or a multiplicative domain decomposition method with an approximate intersecting line preconditioner, for which the number of iterations varies as $m_{k_0} = ch_{k_0}^{-1/2}$.

Let then

$$\tilde{C}_l = (1-\rho)\frac{C_0}{C} c 2^{\alpha l/d'},$$

where

$$d' = \begin{cases} d, & \text{for bisection,} \\ 1, & \text{for congruent divisions,} \end{cases}$$

and let $m_{k_0} = ch_{k_0}^{-\alpha}$, $\alpha = 1$ and $\frac{1}{2}$, in cases (a) and (b), respectively. Substituting this in (3.2) we find

$$\begin{aligned} \frac{W_l}{n_l} &\leqslant \frac{C}{1-\rho}\lambda_0^{1/2}\sigma^{(1/2)(l-k_0)}(1 + \tilde{C}_l 2^{-(\alpha/d')(l-k_0)}\rho^{l-k_0}) \\ &= \frac{C}{1-\rho}\lambda_0^{1/2}2^{((1/2)\log_2\sigma)(l-k_0)}(1 + \tilde{C}_l 2^{-(\alpha/d'+\log_2\rho^{-1})(l-k_0)}), \end{aligned}$$

which we want to minimize. We assume then that

$$\log_2 \rho^{-1} + \frac{\alpha}{d'} > \tfrac{1}{2}\log_2\sigma, \tag{3.3}$$

which holds in the examples to be discussed in the next section, unless σ is excessively large. An elementary computation shows that the minimum is then taken when

$$\tfrac{1}{2}\log_2\sigma + \tilde{C}_l(\tfrac{1}{2}\log_2\sigma - \frac{\alpha}{d'} - \log_2\rho^{-1})2^{-(\alpha/d'+\log_2\rho^{-1})(l-k_0)} = 0,$$

i.e., for

$$l - k_0 = \frac{1}{(d'/\alpha)\log_2\rho^{-1} + 1} l + \tau \tag{3.4}$$

where

$$\tau = \frac{1}{(\alpha/d') + \log_2\rho^{-1}} \log_2 \left((1-\rho)\frac{C_0}{C} c \left(\frac{(2\alpha/d') + 2\log_2\rho^{-1}}{\log_2\sigma} - 1 \right) \right).$$

Hence, since $d' \log_2 \rho^{-1} = d$, condition (3.3) becomes

$$\frac{d+\alpha}{d'} > \tfrac{1}{2}\log_2 \sigma,$$

and (3.4) shows that

$$k_0 = \frac{d}{d+\alpha} l - \frac{d'}{d+\alpha}\tau' \tag{3.5}$$

where

$$\tau' = \log_2 \left[(1-\rho)\frac{C_0}{C} c \left(\frac{2(\alpha+d)}{d' \log_2 \sigma} - 1 \right) \right].$$

Normally, the last term in (3.5) is small, and, at any rate, it is small relative to the first term which grows proportional to the level number of the finest mesh. Hence, it is seen that the asymptotic relation

$$\frac{k_0}{l} = \frac{d}{d+\alpha} - \mathrm{O}(l^{-1}), \quad l \to \infty, \tag{3.6}$$

is essentially independent of σ, which is the only parameter which depends on the shape of the finite elements and the approximations $B_{11}^{(k)}$ used for $A_{11}^{(k)^{-1}}$. Hence, the optimal ratio between the coarsest and finest mesh sizes is essentially independent of these factors. It depends for large l only on the dimension of the problem and on the asymptotic rate of convergence of the coarse mesh solution method.

The minimal computational complexity becomes

$$\frac{W_l}{n_l} \leqslant \frac{C}{1-\rho} \lambda_0^{1/2} \frac{1}{2(\alpha+d)/(d' \log_2 \sigma) - 1} 2^{((1/2)\log_2 \sigma)\left(l/(d/\alpha+1)+(d'/(d+\alpha))\tau'\right)},$$

which can be estimated as

$$\frac{W_l}{n_l} \simeq \text{ const. } h_l^{-(1/2)\log_2 \sigma/(d/\alpha+1)}, \quad \text{as } h_l \to 0 \ (l \to \infty)$$

in case of congruent mesh divisions.

We collect the results of this section in a theorem.

Theorem 3.1. *The total computational complexity of the short FEMLI method when applied for the finite element approximation of* (1.1) *is minimized for the relation*

$$\frac{k_0}{l} = \frac{d}{d+\alpha} - \mathrm{O}(l^{-1}),$$

where k_0 and l are the level numbers of the coarsest and finest meshes, respectively, in a sequence of uniformly refined meshes. The minimal computational complexity per d.o.f. is

$$\frac{W_l}{n_l} \leqslant \textit{ const. } 2^{(l/2)\log_2 \sigma\left(1/(d/\alpha+1)+\mathrm{O}(l^{-1})\right)},$$

Table 1
Values of the stabilization factor for various values of the problem dimension and the rate of convergence parameter α for the coarse mesh solution method

	d	
α	2	3
1	$\frac{1}{3}$	$\frac{1}{4}$
$\frac{1}{2}$	$\frac{1}{5}$	$\frac{1}{7}$

which, in case of congruent triangulation, takes the form

$$\frac{W_l}{n_l} \leqslant \text{const. } h_l^{-(l/2)\log_2 \sigma/(d/\alpha+1)}.$$

Here d is the dimension of the differential equation problem, α, $\alpha > 0$, depends on the complexity of the solution method used for the coarse mesh problem and σ depends on the shape of the finite elements and on the approximation used for $A_{11}^{(k)^{-1}}$. However, σ does not depend on the level number.

It can be seen from (3.2) for the full length FEMLI method where k_0 is fixed, independent of l, that the computational complexity per d.o.f. becomes bounded by $W_l/n_l \simeq \text{const } h_l^{-(1/2)\log_2 \sigma}$.

However, in the optimal short length version of FEMLI, the exponent is stabilized to $\frac{1}{2}\log_2 \sigma/(d/\alpha + 1)$, which, as we shall see in the next section, is a number which grows very slowly with decreasing values of h_l.

We call the factor $1/(d/\alpha + 1)$ the stabilization factor, which takes the values given in Table 1.

4. Examples of finite element meshes

Consider a triangulation of a polygonal domain Ω where the mesh has been constructed to conform with the polygonal shape of the boundary. In case of jumps in the differential equation coefficients one must also adjust the mesh so that no jumps occur inside any element. Each element of this mesh is then divided into four congruent parts and the process is continued until a sufficiently fine mesh has been found.

As shown in [13], the corresponding value of the CBS constant γ_k in (2.8) is then independent of the level number, i.e., $\gamma_k = \gamma$ and γ depends on the angles in the triangulation for the worst triangle. In the case of isosceles triangles (see Fig. 1) and isotropic equation it was shown in [13], (see also [9]) that $\gamma^2 = \frac{1}{2}$. In [6] it was shown that this value holds also for anisotropic problems (and hence for any aspect ratio of the triangles).

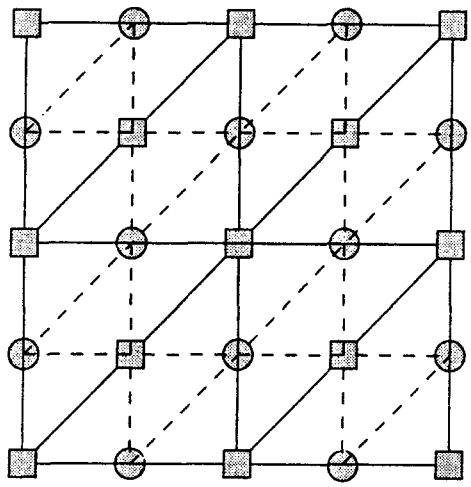

Fig. 1. Congruent mesh division.

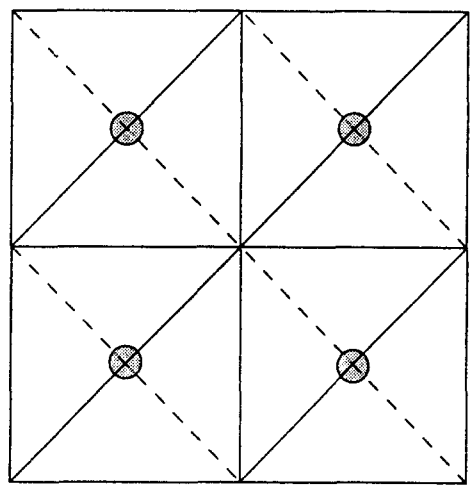

Fig. 2. Bisection of triangles.

The dependence of γ on the angles in the triangulation was derived in [37], where it was shown that

$$\gamma^2 = \frac{3}{4} - \frac{3-f}{2[(4f-3)^{1/2}+3]},$$

where $f = \max f_r$ and $f_r = \sum_1^3 \cos^2 \theta_i^{(r)}$ and $\theta_i^{(r)}$ are the angles in the rth triangle. Hence $\gamma^2 < \frac{3}{4}$ for any triangulation. See also the same reference for some results for tetrahedrons.

For the case of bisections (see Fig. 2) of isosceles triangles for isotropic problems it was shown in [6] that $\gamma^2 = 1/2$, but the value of γ depends heavily on anisotropy and aspect ratios.

Hence if $\alpha_k = 1$ we have $\sigma = 1/(1-\gamma^2) = 2$ for the case of isosceles triangulations and $\sigma < 4$ for any triangulation in the congruent triangulation case.

Now, it follows from (2.7) that $\alpha_k = 1$, among others, in the following cases:

(i) in the AMLI method (see [19]), where

$$A^{(k-1)} = A_{22}^{(k)} - A_{21}^{(k)} B_{11}^{(k)} A_{12}^{(k)},$$

(ii) In the hierarchical basis function method (see[13,40,41]), where

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A^{(k-1)} \end{bmatrix},$$

i.e.,

$$A_{22}^{(k)} = A^{(k-1)}.$$

Clearly, in both cases, it follows by recursion that

$$M^{(k-1)} \geqslant A^{(k-1)} \geqslant A_{22}^{(k)} - A_{21}^{(k)} B_{11}^{(k)} A_{12}^{(k)}.$$

Theorem 3.1 shows then that in the case $\alpha_k = 1$, the computational complexity varies as

$$\frac{W_l}{n_l} \leqslant \mathrm{O}\left(h_l^{-(1/2)\log_2 \sigma}/(d/\alpha+1)\right) = \begin{cases} \mathrm{O}\left(h_l^{-1/(2(2/\alpha+1))}\right), & \text{isosceles triangles, isotropy} \\ \mathrm{O}\left(h_l^{-1/(2/\alpha+1)}\right), & \text{any congruent triangulation, anisotropy} \end{cases}$$

If $\alpha = 1$ we find in the first case

$$\frac{W_l}{n_l} \leqslant \mathrm{O}(h_l^{-1/6}).$$

In the second case it is recommendable to use a more complex coarse mesh solver for which $\alpha = \frac{1}{2}$ holds, in which case

$$\frac{W_l}{n_l} \leqslant \mathrm{O}(h_l^{-1/5}).$$

Note that it follows that for all practical mesh sizes, say $2^{-6} \geqslant h_l \geqslant 2^{-12}$, we have

$$2 \leqslant h_l^{-1/6} \leqslant 4$$

and

$$h_l^{-1/6} \leq \frac{1}{3} \log_2 h_l^{-1}.$$

Hence, it seems that it would only be an academic exercise to try to improve the above results.

For three-dimensional problems, the value of γ^2 is in general bigger but this is compensated by a smaller stabilization factor. In the case when $d = 3$ and $\alpha = \frac{1}{2}$, we have

$$\frac{W_l}{n_l} = \mathrm{O}\left(h_l^{-(1/14)\log_2 \sigma}\right).$$

It can hence be seen that the short FEMLI method stabilizes the recursive two-level methods to a condition number and work per meshpoint which is essentially constant for all practical sizes of mesh. Already in [13] the remarkable stabilization effect was noted for the two-level and three-level FEMLI methods but then only for smaller sized meshes.

An important aspect of the method is the computation of the approximation $B_{11}^{(k)}$ of $A_{11}^{(k)^{-1}}$. In the present paper we shall only shortly comment on this aspect. Consider then the mesh in Fig. 1. Notice that $A_{11}^{(k)}$ can be written as a direct sum of local finite element matrices of order 3×3 which corresponds to node points as indicated. Furthermore, the Schur complement matrix

$$S_1^{(k)} = A_{11}^{(k)} - A_{12}^{(k)} A_{22}^{(k)^{-1}} A_{21}^{(k)}$$

can also be written as a sum of local matrices, in this case of order 6×6, because $A_{22}^{(k)}$ is a diagonal matrix.

Hence, the number δ_k^2 in (2.11) can be computed (as γ_k^2) on a local element by element basis. The computation of $B_{11}^{(k)}$ can be done using the method of approximate inverses, see [7, 34], also for earlier references.

An alternative way is to compute the exact inverses $A_{11,e}^{(k)^{-1}}$ of the local elements $A_{11,e}^{(k)}$ of $A_{11}^{(k)}$ and then let

$$B_{11}^{(k)} = g \sum_e A_{11,e}^{(k)^{-1}}, \tag{4.1}$$

where the summation takes place on all elements. The computed matrix $B_{11}^{(k)}$ must be modified properly to satisfy the assumption (2.3). Among other things the scaling factor g in (4.1) must be chosen so that $B_{11}^{(k)} \leqslant A_{11}^{(k)^{-1}}$. However, we leave the details of the computation of $B_{11}^{(k)}$ to a forthcoming paper.

5. Asymptotically optimal speedup or efficiency

When the FEMLI method is implemented on parallel computers it is important to make the implementation such that interprocessor communication overhead is small.

This topic has been discussed at length in a number of publications, see [8, 16, 20], for instance. The main result is that if one balances the number of processors (p) to the coarsest and finest mesh sizes (n_{k_0} and n_l, respectively) properly, then one can achieve an asymptotically optimal speedup ($=p$) or, equivalently, efficiency ($=1$) as the problem size increases. Here we shall discuss this problem only shortly.

Consider then first the use of Chebyshev iterations for both the outer and inner iterations. Chebyshev iteration is a viable approach because the required eigenvalue information is available using the recursion for the outer iterations and the maximum eigenvalue of the matrix $A^{(k_0)}$ or of $D^{(k_0)^{-1}} A^{(k_0)}$, where $D^{(k_0)} = \mathrm{diag}(A^{(k_0)})$, is readily available. The smallest eigenvalue can be approximated using information about the first eigenvector mode. Alternatively, few steps of an inverse iteration method (where the FEMLI method can be used as solver, see [19]) can be used to estimate the smallest eigenvalue.

The eigenvalue bounds, in particular on the coarsest mesh level can be actually improved using a method of perturbations, see [9, 14], for instance. However, to limit the exposition we will not discuss this topic further here.

To be specific, assume now that the method is implemented on a two dimensional mesh array computer with $\sqrt{p} \times \sqrt{p}$ parallel processors. Then the computing time per iteration of the FEMLI method becomes

$$T_p^{(\mathrm{it})} = \frac{T_1^{(\mathrm{it})}}{p} + w\,\mathrm{it}_{\mathrm{in}} \left(\frac{n_{k_0}}{p}\right)^{(d-1)/d} + w \left(\frac{n_l}{p}\right)^{(d-1)/d} \tag{5.1}$$

Here we have assumed that $p \leqslant n_{k_0}$ so the size of the subgrid mapped on each processor is $m = n_{k_0}/p \geqslant 1$. Naturally, we assume that the mapping is from a domain, decomposed into boxes, which do only need to communicate with the surface points of their $2d$ neighbors.

$T_1^{(\mathrm{it})}$ is the time the computation would have taken on a single processor (including communication time to/from the memory modules), $\mathrm{it}_{\mathrm{in}}$ is the number of inner iterations (on the coarsest mesh), w is a proportionality constant (bandwidth) and $(n_{k_0}/p)^{(d-1)/d}$ is the surface data which must be communicated during one iteration.

Since $T_1^{(\mathrm{it})} = \mathrm{O}(n_l)$, we can neglect the last term in (5.1) (or include it in the first). Clearly, the minimum of

$$\frac{T_1^{(\mathrm{it})}}{p} + w\,\mathrm{it}_{in} \left(\frac{n_{k_0}}{p}\right)^{(d-1)/d}$$

is taken when $p = n_{k_0}$, in which case, using the optimal relation (3.6), i.e., $n_{k_0} \simeq n_l^{d/(d+\alpha)}$, we find

$$T_p^{(\mathrm{it})} \simeq T_1^{(\mathrm{it})}/n_{k_0} + w\,\mathrm{it}_{in} \simeq \mathrm{O}(n_l^{\alpha/(d+\alpha)}) + \mathrm{O}(n_{k_0}^{\alpha/d}) = \mathrm{O}(n_l^{\alpha/(d+\alpha)}).$$

In case $\alpha = 1$ and $d = 2$ we have then

$$T_p^{(\mathrm{it})} = \mathrm{O}(n_l^{1/3}). \tag{5.2}$$

Since, due to the stabilization effect of the short level method, the number of iterations grows only slowly with n_l, the total time is only slightly larger than (5.2), $\mathrm{O}(n_l^{1/3+\tilde{\tau}})$, where $\tilde{\tau}$ is small. Clearly, the speedup, $T_1^{(\mathrm{it})}/T_p^{(\mathrm{it})}$ is asymptotically optimal when $p \leqslant n_{k_0}$.

Consider now the use of the conjugate gradient (CG) method, preconditioned with $M^{(l)}$ for the outer iterations and with some block diagonal method on the coarse level. It is assumed that the preconditioners need only nearest-neighbor processor communication. The computing time per iteration now becomes

$$T_p^{(\mathrm{it})} = \frac{T_1^{(\mathrm{it})}}{p} + w_1\,\mathrm{it}_{in} \left(\frac{n_{k_0}}{p}\right)^{(d-1)/d} + w_2\,\mathrm{it}_{in}\sqrt{p} + w_3\sqrt{p},$$

where the added terms arise from the global communication required for summing up the inner products used in the CG method. For simplicity, consider only the case when $\alpha = 1$, in which case $\mathrm{it}_{in} \sim c n_{k_0}^{1/d}$. Again, the last term can be neglected and we find that $T_p^{(\mathrm{it})}$ is minimized when

$$p = p^* \simeq \left(\frac{T_1^{(\mathrm{it})}}{w_2 c n_{k_0}^{1/d}}\right)^{2/3} \sim \mathrm{const.} n_l^{2d/(3(d+1))},$$

for some constant c. The corresponding computer time is

$$T_p^{(\mathrm{it})} \sim \mathrm{O}(n_l^{(d+3)/(3(d+1))}).$$

If $d = 2$ we find

$$p^* \sim \mathrm{O}(n_l^{4/9})$$

and

$$T_p^{(\mathrm{it})} \sim \mathrm{O}(n_l^{5/9}).$$

The speedup or the efficiency ($E_p = T_1^{(\mathrm{it})}/pT_p^{(\mathrm{it})}$) are not asymptotically optimal in the above case. However, as shown in [18, 20], this can easily be remedied choosing a number of processors which grows slightly slower than the number which minimizes the computing time. For instance $p = p^*/\log n_l$ or even $p = p^*/\log\log n_l$, would be an efficient choice.

That the above theoretical findings are practically viable is seen for typical values of n_l. Take $n_l = 2^{9d}$, for instance. Then

$$p \sim \text{ const.}2^{6d^2/(d+1)} \simeq \begin{cases} 2^8 = 256, & \text{if } d = 2, \\ 2^{13.5} \simeq 11.10^3, & \text{if } d = 3. \end{cases}$$

Note that the conclusion of the above results is that one should not use too many processors, but any number $p < p^*$ is efficient.

6. Conclusions

It has been shown that by balancing the size of the coarsest mesh to the finest and the number of processors to these mesh sizes properly one can achieve a method of V-cycle FEMLI form which has both an essentially constant computational complexity per mesh point for all practical mesh sizes and an asymptotically optimal speedup or efficiency. These results have been shown to hold for elliptic problems with anisotropy and arbitrary jumps of the coefficients in the differential equations between the elements of the coarsest mesh, which, by itself, is quite a fine mesh.

The above promises to solve a long-standing open problem, namely how to get parallel efficiency in massively parallel computation out of multilevel iteration methods. For a survey and critics of earlier attempts, see [38], for instance.

Finally, let us mention that since many more general problems can be rewritten as a sequence of elliptic solvers, so the FEMLI method is applicable also in such contexts. For instance, it has been shown recently that it can be used efficiently in a Stokes problem solver for incompressible flows, see [10, 20].

References

[1] O. Axelsson, On multigrid methods of the two-level type, in: W. Hackbusch and U. Trottenberg, Eds., *Proc. on Multigrid Methods*, Köln-Porz, 1981, Lecture in Notes Math. **960** (Springer, Berlin, 1982) 352–367.

[2] O. Axelsson, A general incomplete block-matrix factorization method, *Linear Algebra Appl.* **74** (1986) 179–190.

[3] O. Axelsson, A multilevel solution method for nine-point difference approximations, in: G.F. Carey, Ed., *Proc. on Parallel Supercomputing: Methods, Algorithms and Applications*, (Wiley, New York, 1989) 191–205.
[4] O. Axelsson, An algebraic framework for hierarchical basis function multilevel methods or the search for "optimal" preconditioners, in: D.R. Kincaid and L.J. Hayes, *Iterative Methods for Large Linear Systems*, Proc. Conf. on Iterative Methods, Austin, TX, 1988 (Academic Press, NewYork, 1990) 17–40.
[5] O. Axelsson, The method of diagonal compensation of reduced matrix entries and multilevel iterations, *J. Comput. Appl. Math.* **38** (1991) 31–43.
[6] O. Axelsson, On Algebraic Multilevel Iteration Methods for Selfadjoint Elliptic Problems with Anisotropy, Rend. Sem. Mat. Univer. Politech. Torino, Fascicolo Speciale *Numerical Methods*, (1991), 31–61.
[7] O. Axelsson, *Iterative Solution Methods* (Cambridge University Press, New York, 1994).
[8] O. Axelsson, The algebraic multilevel iteration method: a scalable and optimal algorithm, in: W.G. Habashi Ed., *Solution Techniques for Large-Scale CFD Problems* (Wiley, New York, 1995) 137–160.
[9] O. Axelsson, V.A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computation* (Academic Press, Orlando, Florida, 1984).
[10] O. Axelsson, V.A. Barker, M. Neytcheva and B. Polman, Solving the Stokes problem on a massively parallel computer, submitted to Numerical Methods in Fluids.
[11] O. Axelsson, S. Brinkkemper and V.P.II'in, On some versions of incomplete blockmatrix factorization iterative methods, *Linear Algebra Appl.*, **58** (1984) 3–15.
[12] O. Axelsson and V. Eijkhout, The nested recursive two-level factorization method for nine-point difference matrices, *SIAM J. Sci. Statist. Comput.* **12** (1991), 1373–1400.
[13] O. Axelsson and I. Gustafsson, Preconditioning and two-level multigrid methods of arbitrary degree of approximation, *Math. Comp.* **40** (1983) 219–242.
[14] O. Axelsson, Yu. R. Hakopian and Yu. A. Kuznetsov, Multilevel preconditioning for perturbed finite element matrices, IMA Journal of Numerical Analysis, to appear.
[15] O. Axelsson and G. Lindskog, Constant wavefront iteration methods for nine- and 15-point difference matrices, *Computing* **46** (1991) 233–252.
[16] O. Axelsson and M. Neytcheva, Scalable algorithms, for the solution of Navier's equations of elasticity, *J. Comput. Appl. Math.* **63** (1995) 149–178.
[17] O. Axelsson and M. Neytcheva, Parallel implementations of the algebraic multilevel iteration method, in: J.G. Lewis, Ed., *Proc. 5th SIAM Conf. on Applied Linear Algebra* (SIAM, Philadelphia, 1994) 372–376.
[18] O. Axelsson and M. Neytcheva, The short AMLI method-II. Computational and communication Complexity, in: I. Dimov et al. Eds., *Proc. 3rd Internat. Conf. on Advances in Numerical Methods and Applications* $O(h^3)$, Sofia, Bulgaria 1994, 162–169.
[19] O. Axelsson and M. Neytcheva, Algebraic multilevel iteration method for Stieltjes matrices, *Numer. Linear Algebra Appl.* **1** (1994), 213–236.
[20] O. Axelsson and M. Neytcheva, *Scalable Parallel Algorithms in CFD Computations, Computational Fluid Dynamics Review, 1995* (Wiley, New York, 1995) 837–857.
[21] O. Axelsson and P. Vassilevski, A survey of multilevel preconditioned iterative methods, *BIT* **29** (1989) 769–793.
[22] O. Axelsson and P. Vassilevski, Algebraic multilevel preconditioning methods I, *Numer. Math.* **56** (1989) 157–177.
[23] O. Axelsson and P. Vassilevski, Algebraic multilevel preconditioning methods II, *SIAM J. Numer. Anal.* **27** (1990) 1569–1590.
[24] O. Axelsson and P. Vassilevski, Asymptotic work estimates for AMLI methods, *Appl. Numer. Math.* **7** (1991) 437–451.
[25] N.S. Bakhvalov, On the convergence of a relaxation method with natural constraints on the elliptic operator, *U.S.S.R. Comput. Math. Math. Phys.* **6** (1966) 101–135.
[26] R. Bank and T. Dupont, Analysis of a two-level scheme for solving finite element equations, Report (NA-1559), Center for Numerical Analysis, Univ. Texas at Austin, 1980.
[27] J.H. Bramble, *Multigrid Methods*, Pitman Res. Notes in Math. **294**, (Longman, New York, 1993)
[28] A. Brandt, Multi-level adaptive solution to boundary-value-problems *Math. Comp.* **31** (1977), 333–390.
[29] P. Concus, G. Golub and G. Meurant, Block preconditioning for the conjugate gradient method, *SIAM J. Sci. Statist. Comput.* **6** (1985) 220–252.

[30] R.P. Fedorenko, A relaxation method for solving elliptic difference equations, *U.S.S.R. Comput. Math. and Math. Phys.* **1** (1962) 1092–1096.
[31] R.P. Fedorenko, The speed of convergence of one iteration process, *Zh. Vycisl. Mat. i Mat. Fiz.* **4** (1964) 559–563.
[32] W. Hackbusch, Multigrid convergence theory, in: W. Hackbusch and U. Trottenberg, Eds., *Proc. on Multigrid Methods,* Köln-Porz, 1981, Lecture Notes in Math. **960** (Springer, Berlin, 1982) 177–219.
[33] W. Hackbusch, *Multigrid Methods and Applications* (Springer, Berlin, 1985).
[34] L.Yu Kolotilina and A. Yeremin, Factorized sparse approximate inverse preconditionings, *SIAM J. Matrix. Anal. Appl.* **14** (1993) 45–58.
[35] Yu.A. Kuznetsov, Multigrid domain decomposition methods for elliptic problems, in: *Proc. 8th Internat. Conf. on Computational Methods for Applied Science and Eng.,* Vol. **2**, (1987) 605–616.
[36] Yu.A. Kuznetsov, Algebraic multigrid domain decomposition methods, *Soviet. J. Numer. Anal. Math. Modelling* **4** (1989) 351–379.
[37] J.F. Maitre and F. Musy, The contraction number of a class of two-level methods; an exact evaluation for some finite element subspaces and model problems, in: W. Hackbusch and U. Trottenberg, Eds., *Proc. on Multigrid Methods,* Köln-Porz, 1981, Lecture Notes in Math. **960** (Springer, Berlin, 1982) 535–544.
[38] L.R. Matheson and R.E. Tarjan, A critical analysis of multigrid methods on massively parallel computers, in: N.D. Melson, T.A. Manteuffel and S.F. McCormick, eds., *Proc. 6th Copper Mountain Conf. on Multigrid Methods,* (NASA 1993) 405–421.
[39] P.S. Vassilevski, Multilevel preconditioning matrices and multigrid V-cycle methods, in: W. Hackbusch, Ed. *Proc. 4th GAMM-Seminar*, Kiel, 22–24 January 1988, Notes on Numerical Fluid Mechanics **23** (Vieweg, Braunschweig, 1988) 200–208.
[40] P. Vassilevski, Hybrid V-cycle algebraic multilevel preconditioners, *Math. Comp.* **58** (1992) 489–512.
[41] H. Yserentant, On the multilevel splitting of finite element spaces, *Numer. Math.* **49** (1986), 379–412.